INTRODUCTION TO SPEECH RECOGNITION

Mini Project

Article Chosen: "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition" by Pete Warden

Prepared by: Group 8

Name	Matric No
Khairol Izzul Firdaus Bin Khairol Hisam	**A21EC0036**
Adam Azhar Bin Nor Adha	**A21EC8010**
Muhammad Muadz bin Jamain	**B22EC0032**
Muhamad Adib Wafi Bin Muhamad Jais	**A21EC0056**

Video Presentation Link:

https://drive.google.com/drive/folders/1efapSHMiXVSntTlWoe8xQzWoZ3ZKHixk?usp=sharing

Table of Content

1.0 Introduction

These days, it’s common to talk to our phones or smart devices. We say things like “Hey Google” or “Stop the music” and expect a quick response. For that to happen, the device needs to recognize short voice commands. This process is called keyword spotting which means picking out certain words from speech.

Many speech recognition tools are built to handle full conversations or long sentences. But for simple commands, that’s often too much. We need smaller systems that can run on phones or home devices without needing the internet or lots of power.

The author of the article "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition", Pete Warden from Google Brain noticed this problem and he created the Speech Commands Dataset to solve this exact problem. His goal was to give people a free and easy-to-use dataset focused only on short, one-second voice commands like “yes”, “no” or “go.” By doing this, he hoped to make it easier to train lightweight speech models that could run on devices with limited memory and power.

This paper explains how the dataset was built, how it can be used for keyword spotting and how it helps improve voice-controlled systems especially in real-world environments with noise or different accents.

2.0 Problem Statement and Scope of the Article

2.1 Problem Statement of Article

As voice control becomes more common in everyday life, many speech recognition systems are still not suitable for small or affordable devices. Most of these systems are designed to process full sentences and often require powerful computers to function well. However, in real-world situations like using a smart speaker or a basic mobile app. we often just need to detect a short word such as “yes,” “no,” or “stop.”

The problem is, there wasn’t a good, publicly available dataset focused on short speech commands. Developers and researchers who wanted to build lightweight systems had limited resources to train their models. Without the right kind of data, it’s difficult to create accurate, reliable systems that can handle background noise, different speakers and realistic settings.

2.2 Scope of the Article

This article focuses on solving that issue by introducing the Speech Commands Dataset. The dataset contains thousands of one-second audio clips with people saying common words. These recordings come from various speakers using regular microphones in everyday environments, making the data more practical and realistic.

The main goal of this dataset is to help people build simple, efficient speech models that can recognize short commands accurately. These models are meant to run on devices with limited memory and processing power. The paper also explains how the dataset can be used to train and test models fairly, giving researchers and developers a shared benchmark for comparison.

3.0 Main Methods and Formulations

The main focus of this paper is the creation and preparation of a speech dataset that can help train models to recognize short voice commands. This dataset is called the Speech Commands Dataset, and it includes a total of 105,829 one-second audio clips recorded by more than 2,600 different people. These clips include common words like “yes,” “no,” “go,” and “stop,” as well as background noise samples.

To make the dataset usable for machine learning, several important steps were taken:

1. **Filtering**: Clips that were too short or corrupted (less than 5KB) were removed to keep the dataset clean.
2. **Trimming and Normalizing:** Audio files were trimmed so that the loudest part of the word is centered. This makes it easier for models to detect speech correctly.
3. **Manual Labeling and Verification:** Each clip was labeled with the word that was spoken. Volunteers helped confirm that each label matched what was actually said.
4. **Folder Structure:** Clips were sorted into folders named after each spoken word like “yes”, “no”, etc. to make it easy to load them in code.
5. **Background Noise Clips:** The dataset also includes clips with only noise such as pink noise, white noise and background talking. These help train the model to ignore irrelevant sounds.

Figure 1 shows the overall process of preparing the Speech Commands Dataset, starting from recording voice samples, followed by filtering out damaged files, trimming and centering the audio, verifying labels, organizing the data into folders and finally using it to train a machine learning model.

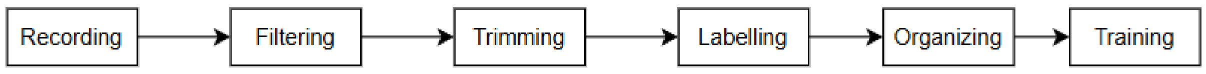


Figure 3.1: Data Preparation Pipeline for the Speech Commands Dataset

To make use of this dataset, the paper tested a simple machine learning model called a Convolutional Neural Network (CNN). Since CNNs are commonly used for image recognition, the audio clips were first converted into spectrograms which are visual representations of sound over time. These spectrograms show how the sound's frequency changes, similar to heatmaps.

The model was tested using a single method called Top-One Accuracy, where the goal is to correctly identify which one word was said out of 12 possible options. This kind of test is useful for small devices that only need to recognize basic voice commands. The paper did not perform real-time or continuous listening tests but the author mentioned that this type of streaming detection would be important in the future.

Table 1: Model Testing Summary

Test Type	Description	Result
Top-One Accuracy	Predict one word out of 12 options	88.4%
Streaming Test	Real-time voice detection in continuous audio	Not tested

This method allows developers to train speech recognition models that can run on small devices and recognize short commands accurately. The paper's approach keeps everything simple, affordable, and practical for real-world applications.

4.0 Discussion on the Results of the Article

The results in this paper focus on how well a basic speech recognition model performs using the new Speech Commands Dataset. The author tried out a basic model known as a convolutional neural network (CNN), which listens to short one-second audio clips, changes them into visual patterns called spectrograms, and then figures out which word was spoken from a list of 12 options like “yes”, “no”, “stop” and “go”. Figure 4.1 shows a simplified CNN pipeline showing input audio, spectrogram generation, CNN processing, and final prediction.

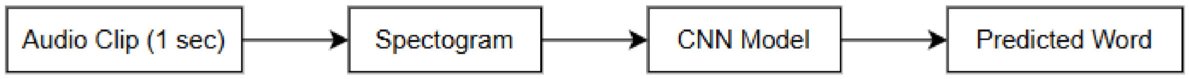


Figure 4.1: Basic CNN speech recognition pipeline

A key finding from this test is that the model got the correct answer around 88.4% of the time. That means it recognized the right word in nearly nine out of ten tries. This is a good sign, especially since the model was designed to be small and light like the kind that could run on mobile phones or other devices with limited power and memory.

To help the reader understand what the model sees, we can also show a sample spectrogram. This is the image version of sound, which the CNN uses to recognize patterns.

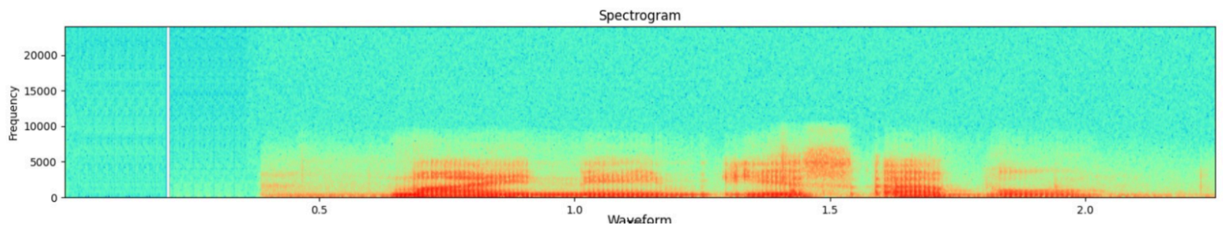


Figure 4.2: Sample Spectrogram of audio clip

One of the reasons the model did well is because the dataset was made carefully:

- Well-organized with each word stored in separate folders

- Clean and labeled, thanks to manual verification
- Diverse with over 2,600 different speakers

The paper also highlights that the model was able to ignore background noise quite effectively, especially because of the inclusion of noise-only clips during training. These extra noise samples help the model learn what “not a command” sounds like, which reduces false triggers.

However, the results are limited to the specific test setup. The model was only tested on pre-recorded clips, not on real-time or streaming audio. This means the results show that the model works well under controlled conditions but may need further improvement to perform reliably in real-world use.

Still, for a first release of a dataset and a baseline model, achieving 88.4% accuracy is a strong indicator that the dataset is useful and that the approach is practical for many real-life speech command applications.

5.0 Shortcomings and Suggested Improvements

5.1 Shortcomings of the Article

While the paper shows good progress with a simple CNN model, there are several limitations in the current approach. First, the model was only tested on pre-recorded one-second audio clips, not real-time audio or streaming speech. This makes it difficult to tell if the model would perform well in real-world use such as in smart speakers or mobile apps where users speak naturally. Real-time speech involves background noise, different accents, changes in pitch and non-stop input (things the model hasn't been tested against).

Another limitation is that the system is trained only to detect short, fixed words from a limited set like "yes", "no" or "stop". It cannot recognize longer phrases, natural language or new commands that were not part of the training set. This restricts how flexible or intelligent the system can be when used in more advanced situations.

In addition, the model depends on manual feature extraction through spectrograms and although the CNN can recognize patterns, it may still struggle when sounds are unclear or overlap. These limitations suggest that the model is best used as a basic tool, not a fully capable speech assistant.

5.2 Suggested Improvements Using Gemini API

One promising improvement is to use the Gemini API like Google's Gemini 1.5 or Gemini 2.0 models. It can support multimodal and context-aware AI processing. Unlike traditional CNNs, Gemini models can:

- Understand full sentences and context
- Recognize varied speech patterns including different accents, noise levels or real-time speech
- Generate or classify language more flexibly using advanced large language model (LLM)
- Work without needing to convert audio to spectrograms manually

Table 5.1 shows how the Gemini API could improve this paper’s method:

Table 5.1: Limitation and Improvement of the Article

Limitation in Original Paper	**Improvement with Gemini API**
Only works on 1-second fixed clips	Gemini can work with longer and real-time audio
Only detects 12 hardcoded commands	Gemini can handle natural language and new commands
Needs manual spectrogram conversion	Gemini can take raw audio directly as input
Can’t understand context or follow-up speech	Gemini can track conversational flow and intent

With Gemini, it would be possible to build a more intelligent speech assistant that not only recognizes short keywords but also understands commands in natural, real-time conversations and works efficiently on edge devices like phones or IoT gadgets. This would transform the original dataset and model into a next-generation voice system suitable for modern smart devices, educational tools and more.

6.0 Working Demo

6.1 Overview of the Prototype

Our working demo is a real-time voice recognition and interpretation system that builds upon the concepts discussed in Pete Warden's paper but extends them significantly with modern AI capabilities. Unlike the original Speech Commands Dataset approach which focused only on recognizing isolated, one-second commands from a limited vocabulary, our prototype implements a complete end-to-end solution that captures continuous speech, transcribes it in real-time and processes it through Google's Gemini AI for intelligent interpretation. The system is built as a Node.js web application with a responsive frontend that provides immediate visual feedback to users. It uses the Web Speech API for client-side speech recognition, communicates with the server via Socket.io for real-time data exchange and integrates with Google's Gemini API for advanced natural language understanding and contextual responses.

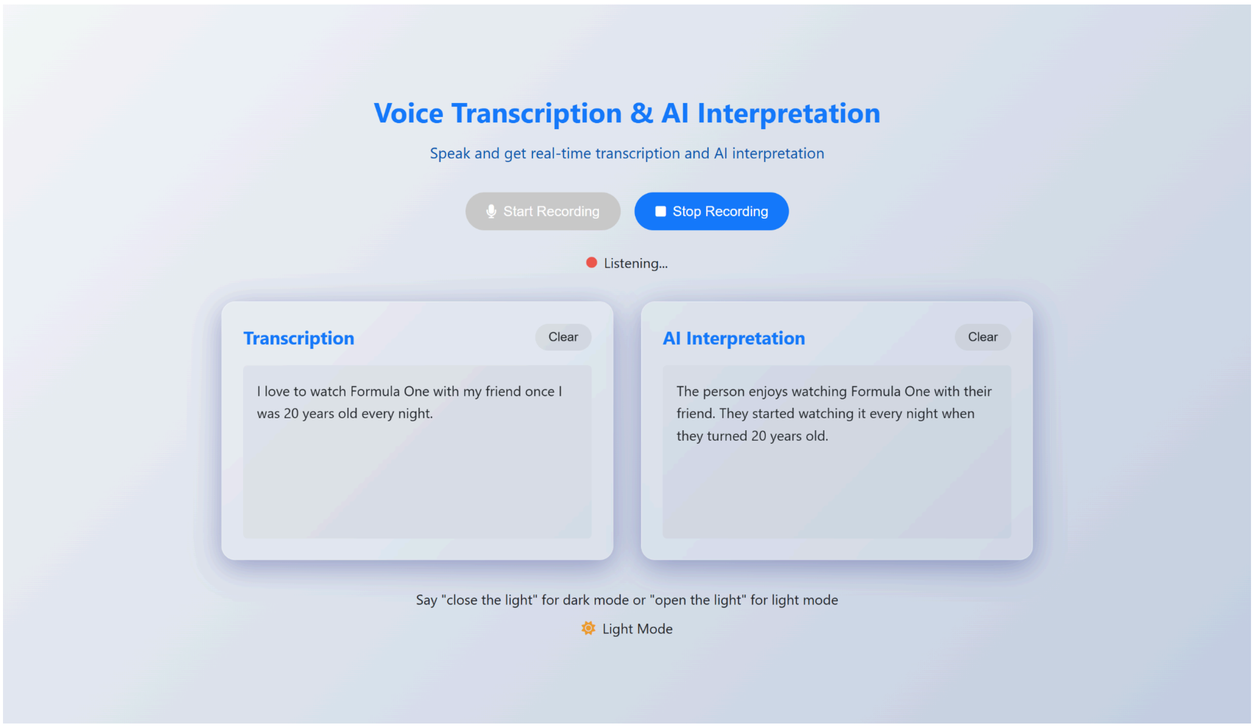
.

6.2 Key Features

Table 6.1: Key Features

Feature	**Description**
Continuous Speech Recognition	Unlike the original paper which used one-second audio clips, this system processes continuous speech in real-time. This allows for a more natural conversation flow.
Real-time Transcription	Spoken words are instantly converted to text using the browser’s built-in Speech Recognition API. There’s no need for manual processing or spectrogram generation like in the original paper.
Advanced AI Interpretation	Instead of just classifying speech into 12 categories with 88.4% accuracy like the original CNN model, this system uses Google’s Gemini AI to understand the full meaning of natural language and give relevant responses.
Voice-Activated UI Controls	Users can use speech commands like "close the light" or "open the light" to switch between dark and light modes. This shows how voice can control interface elements.

6.3 Screenshot of the Working Demo

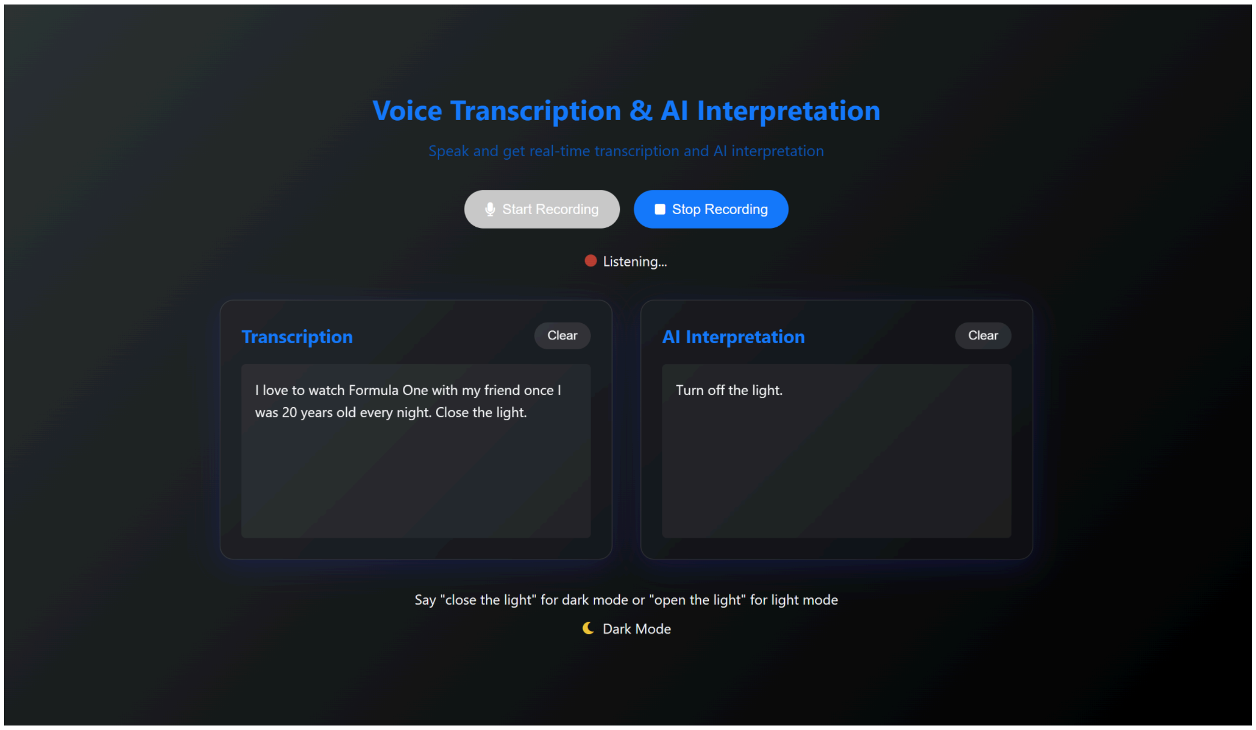

Voice Transcription & AI Interpretation
Speak and get real-time transcription and AI interpretation
Start Recording
Stop Recording
Listening...
Transcription
Clear
AI Interpretation
Clear

I love to watch Formula One with my friend once I was 20 years old every night.

The person enjoys watching Formula One with their friend. They started watching it every night when they turned 20 years old.

Say "close the light" for dark mode or "open the light" for light mode
Light Mode

Figure 6.1: Working Demo Page (light mode)


Voice Transcription & AI Interpretation
Speak and get real-time transcription and AI interpretation
Start Recording
Stop Recording
Listening...
Transcription
Clear
AI Interpretation
Clear

I love to watch Formula One with my friend once I was 20 years old every night. Close the light.

Turn off the light.

Say "close the light" for dark mode or "open the light" for light mode
Dark Mode

Figure 6.2: Working Demo Page (dark mode)

6.4 How It Improves Upon the Research Paper

Our working demo overcome several key limitations identified in Pete Warden's original paper:

1. Beyond Limited Vocabulary: While the Speech Commands Dataset was restricted to just 12 words, our system can understand and process natural language in its entirety, enabling more complex and meaningful interactions.
2. Real-time Processing: Unlike the paper's approach which only worked with pre-recorded clips, our system processes speech in real-time, making it practical for actual user interactions.
3. Contextual Understanding: The original CNN model could only classify isolated words without understanding context. By integrating Gemini AI, our system comprehends the semantic meaning behind user utterances and responds appropriately.
4. No Manual Feature Engineering: The paper relied on manual conversion of audio to spectrograms. Our system eliminates this step by using modern speech recognition APIs that handle the audio processing internally.
5. Practical Application: We've demonstrated a practical use case (theme switching) that shows how voice commands can be integrated into real applications, moving beyond the theoretical approach of the original paper.
6. Continuous Improvement: Unlike the static model described in the paper, our integration with cloud-based AI services means the system can benefit from ongoing improvements to the underlying models without requiring retraining.

This working demo represents a significant advancement over the original research, showing how much modern AI technologies can transform the limited-vocabulary speech recognition into a powerful, context-aware voice system suitable for real-world applications.

6.5 Prototype Resources

GitHub Repository: https://github.com/muadzjamain/speech_recognition_project

Live Demo: https://speechproject-group8.netlify.app

Video Link:

https://drive.google.com/drive/folders/1efapSHMiXVSntTlWoe8xQzWoZ3ZKHixk?usp=sharing

7.0 Conclusion

This paper introduces the Speech Commands dataset which provides a valuable foundation for building simple, low-cost voice recognition systems. The main goal was to help developers, students and researchers create models that can recognize short spoken words like "yes", "no" or "stop" using small, efficient neural networks such as CNNs. The dataset was well-prepared with clean labels and wide voice diversity and the basic model tested achieved a strong 88.4% accuracy rate.

However, due to the limitations of the current model, it only works on short, pre-recorded clips and doesn't handle real-time speech or longer conversations. It also isn't designed to understand context. Because of these limits, the system is more of a stepping stone than a complete solution.

Tools like Google's Gemini API could help take this work further. Gemini models are built to process raw audio, follow conversation flow and handle real-time input. This means they could offer a more practical and advanced solution for speech recognition. In short, the paper lays a strong foundation for basic voice control and with newer AI tools, it has the potential to grow into something much more powerful for real-world use.

8.0 Reference

[1] Warden, P. (2018). *Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition*. arXiv preprint arXiv:1804.03209. https://doi.org/10.48550/arXiv.1804.03209

[2] Google. (2024, February). *Introducing Gemini 1.5: Advancing multimodal understanding*. Google Blog. https://blog.google/technology/ai/google-gemini-next-generation-models

[3] TensorFlow. (n.d.). *Simple audio recognition: Recognizing keywords*. TensorFlow. https://www.tensorflow.org/tutorials/audio/simple_audio